Ansys

WHITE PAPER

SysML Behaviors for Mission Decision

Authors: Noah Ingwerson, John Vergere, Johnathan Thompson, Shashank Narayan

Abstract *— Systems Modeling Language (SysML) is an industry standard for defining system architectures. SysML-based architectural designs can also be used to model entire missions. These designs must be validated to ensure that they accurately reflect the complexity of the mission and its real-world operating environment, and that mission objectives are met. An effective way to accomplish this is to execute the SysML models of a system in a full physics-based simulation of its operating environment. In addition, SysML behavior models in a simulation environment can be used to establish and evaluate correlations between transitions in SysML workflows and those same events within realistic simulations. By predicting outcomes, these capabilities can be used to provide effective decision support for entire missions.*

/ Contents

Introduction **2**
- MBSE Overview 2
- SysML Overview 2
- Mission Overview 2

Models **3**
- Structure 3
  - SARCommandControl 4
  - FishingVessel 4
  - LEODetectingConstellation 4
  - AutonomousSARUAV 4
  - MannedSARPlatform 4
- Behavior 5
  - FishingVessel State Machine Diagram 5
  - LEODetectingConstellation State Machine Diagram 5
  - AutonomousSARUAV State Machine Diagram 6
  - MannedSARPlatform State Machine Diagram 7
  - SARCommandControl State Machine Diagram 8
  - Instance Specification Diagram 9
- Analysis Models 9

Execution **10**
- Initialization 10
- Simulation 11
- Validation 11

Conclusion **12**

Introduction

This paper will explore the execution of a SysML model and demonstrate how this capability can be used to validate designs against a set of requirements. This notional example focuses on a search-and-rescue (SAR) mission that requires multiple disparate systems to function cohesively through coordination and communication to ensure a successful outcome. While the example demonstrated here is relatively simple, the same methodology can be applied to any real-world system designed with model-based systems engineering (MBSE).

Before we can begin this exploration, however, we need to define the key concepts involved — as they pertain to our modeling and analysis in this instance. These concepts are model-based systems engineering (MBSE), Systems Modeling Language (SysML), and the mission model that we will use to represent the operational requirements and environment of the system we are modeling.

MBSE Overview

Model-based systems engineering is a methodology that focuses on creating digital system and engineering domain models and using them as the primary means of exchange of information, feedback, and requirements — as opposed to document-centric systems engineering. MBSE involves the entire process of capturing, communicating, and ensuring that all digital models used to represent a system are coordinated and maintained throughout the entire life cycle of the system.

MBSE was developed to replace static documents with dynamic views of digital models that provide interactive insight into the important metrics of a system under design. Instead of documents that are at best organized into folders, these digital models are connected by a "digital thread" that can be followed to understand the entire design.

SysML Overview

SysML is one language that enables MBSE. It was developed by the Object Management Group (OMG) and International Council on Systems Engineering (INCOSE) with the **OMG SysML 1.0** specification.

The key elements of SysML that are used in the model we will explore in this paper are described below.

- **Package:** A package is like a folder on your operating system. It is a container used to organize elements of the model into separate collections.
- **Block:** A block is the lowest level unit of structure and is like a blueprint for an object. The block describes the object using properties, operations, and ports. Properties can be simple value types or other blocks that represent important attributes of the block. Operations are actions that the block can perform, with input arguments, and outputs returned by the operation.
- **Block definition diagram** (BDD): A block definition diagram visually displays the relevant characteristics of the blocks and the relationships between them. SysML diagrams only show one view of the model and don't necessarily encompass every aspect of the model.

  An object diagram shows instances of blocks. If the block is a blueprint for an object, the instances are the actual representations of the individual objects. There can be multiple instances of any given block. Instances can be assigned values to each of the properties defined on the block.
- **State machine diagram:** A state machine diagram is composed of blocks that represent the possible states that a system can be in. States will have directional arrows that indicate the flow from state to state, when the transition between states occurs, and what happens because of that transition.

Mission Overview

The mission that is described in this model is a notional search and rescue mission involving many systems that must communicate and coordinate with each other to reach a successful outcome. The main systems in this mission are:

- A fishing vessel
- Low Earth orbit (LEO) satellite constellation
- Command and control (C2) node
- Autonomous uncrewed aerial vehicle (UAV)
- Crewed search and rescue platform

At the start of the mission, the fishing vessel is operating at sea and loses a person overboard during stormy conditions. The overall goal is to locate the person overboard and initiate a recovery with the crewed search and rescue platform before too much time has passed. To do this, each system must be able to perform its given task while maintaining communication with the command and control node.

After losing the person overboard, the fishing vessel restores its communication capability as the storm passes. When able, it emits an alert to notify assets in communication range that a person has been lost overboard.

The LEO satellite constellation comprises satellites spread across multiple orbital planes. Each satellite is equipped with a communication payload that enables it to communicate with locations on the ground from orbit. For simplicity, it is assumed that the satellites are able to communicate with any system that is in direct line of sight. The role of the satellite constellation is to receive distress signals from the fishing vessel and relay that transmission to the command and control node. The satellite that receives the distress signal is also responsible for identifying an initial area for operations by performing a wide area geolocation of where the signal came from.

The autonomous UAV is a small-to-mid-size platform that is equipped with a sensing payload — to detect the person overboard — and communication equipment. The same simplification for satellite communication applies to the UAV. In this analysis, the details and phenomenology of the sensor are not modeled. Instead, the sensor is given a geometric field of view that represents its effective detection volume. The role of this UAV is to perform a predefined search of the area outlined by the satellite constellation and attempt to locate the person overboard. If a person is detected with the onboard sensor, the UAV relays the location of the person to the C2 node and maintains a holding pattern above them.

The crewed search and rescue platform for this mission is a medium-range helicopter that can support a crew of four. It is also equipped with a communication payload to coordinate with the C2 node and receive information from the UAV. It can be dispatched with the UAV to wait in the area until the person overboard is located, or it can be sent to the person's location after detection by the UAV. The crewed SAR platform is responsible for recovering the person overboard and returning them to base where medical aid can be given.

To reach a successful mission outcome, each system must be capable of performing its role and responsibilities and working in a coordinated environment to recover the person overboard within a given amount of time. The mission is executed in a variety of operational conditions that will stress the capabilities of each system to identify any problematic situations that could cause mission failure.

Models

Structure

Each of the five main systems for this mission are defined as blocks in the block definition diagram (Figure 1). For simplicity, the components, subsystems, and payloads for these systems are represented as properties on each block instead of further decomposing the model.

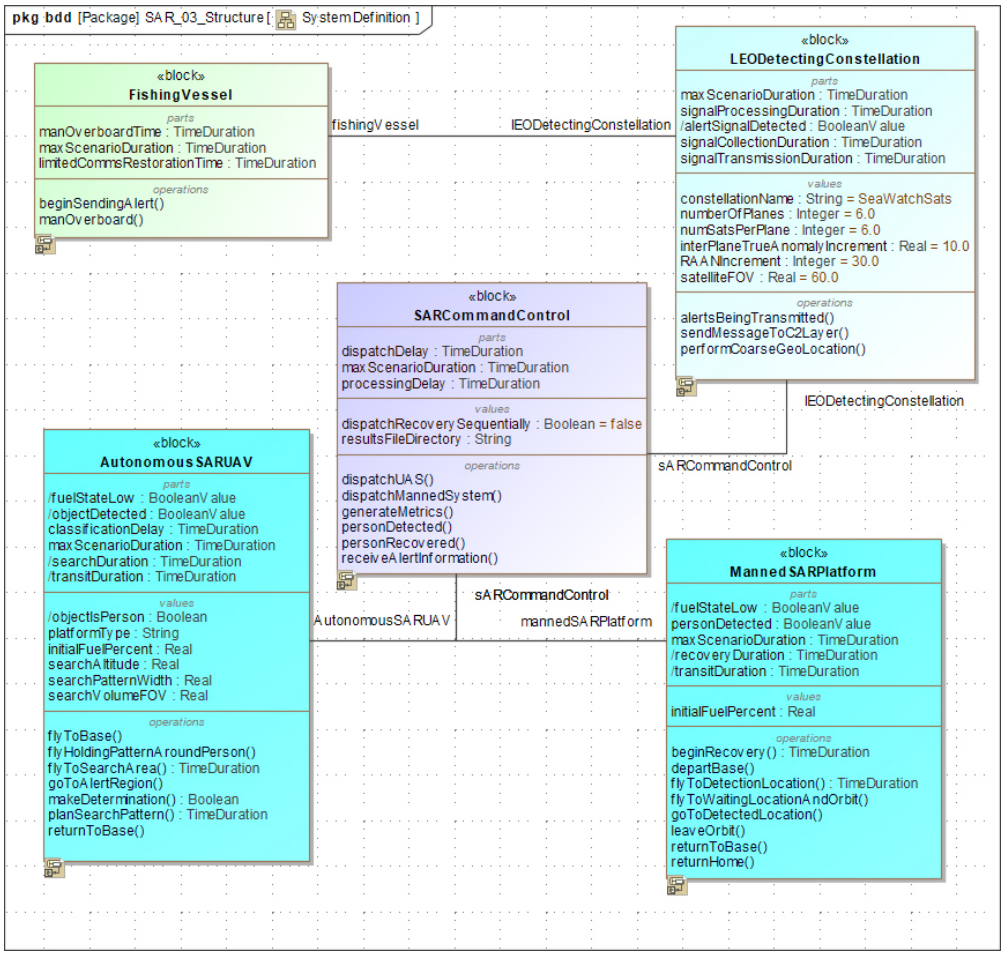

Figure 1. Block definition diagram for the mission

SARCommandControl

At the center is the SARCommandControl block that defines the structure of the C2 node. The SARCommandControl block has a bidirectional association with all the other systems in the mission: the LEODetectingConstellation, AutonomousSARUAV and MannedSARPlatform. These associations are used to share data between the systems during the mission.

The “dispatchDelay” and “processingDelay” part properties on SARCommandControl are abstractions to represent the time it takes internal components to perform given operations or for signals to travel. In a real system with the internal components fully modeled, these properties would not be necessary. The “dispatchRecoverySequentially” is a Boolean value property that is used in the state machine diagram to determine the flow of operations.

Finally, the operations listed under the SARCommandControl are used to directly interact with the other systems. They will either send signals to the specified system (dispatchUAS) or be called by other systems through the association relationship to trigger a call event (personDetected).

FishingVessel

The other blocks are set up similarly. The fishing vessel has an association to the LEODetectingConstellation since the constellation will receive the distress signal from the vessel and attempt to locate it. The “limitedCommsRestorationTime” is a set amount of time taken for the vessel to restore its communication capability after the storm. The “manOverboardTime” is the point in time when the person goes overboard. In the model, it can be given an explicit time or stochastically generated for execution of the mission. The only operations available to the FishingVessel are “manOverboard,” which sends the person overboard, and “beginSendingAlert,” which starts the transmission of the distress signal.

LEODetectingConstellation

The LEODetectingConstellation block represents the collection of satellites, with the value properties “numberOfPlanes” and “numSatsPerPlane” used to specify the total number of satellites in the constellation. The “RAANIncrement” and “interPlaneTrueAnomalyIncrement” determine the spacing of the constellation, while the “satelliteFOV” determines the viewing geometry of each satellite.

Like the other blocks, most of the part properties are internal delays and signal transmission times that will be used for time event triggers in the state machine diagram. The LEODetectingConstellation also has an “alertSignalDetected” property, which is a Boolean value. This value type can be thought of as a time-dynamic Boolean. Before the fishing vessel loses a person overboard and begins transmitting an emergency alert, this value will be false. When a satellite in the constellation receives the alert, this value will switch to true. This is used to drive transitions in the behaviors.

The “sendMessageToC2Layer” and “performCoarseGeoLocation” operations are actions that can be performed by the satellite constellation, while the “alertsBeingTransmitted” operation is used as a call event.

AutonomousSARUAV

The AutonoumousSARUAV has value properties to define the characteristics of its search pattern, as well as the type of aircraft and amount of fuel it is carrying at takeoff. Under the part properties, there are Boolean properties “fuelStateLow” and “objectDetected” that are used to drive behaviors in the state machine diagram, like the “alertSignalDetected” property on the LEODetectingConstellation block. The “classificationDelay” property represents internal processing when the AutonomousSARUAV detects an object, and the “searchDuration” and “transitDuration” properties represent the flight time to the search area and the search itself respectively.

The “makeDetermination” and “planSearchPattern” operations on the AutonomousSARUAV are actions performed by the computer. “makeDetermination” will identify an object detected by the onboard sensor as a person or not while “planSearchPattern” will create a raster search pattern based on the pattern value properties described above and compute the amount of time needed to fly the route. The rest of the operations control the flight of the vehicle.

MannedSARPlatform

The MannedSARPlatform has very similar properties to the AutonomousSARUAV — Boolean-value properties for behavior transitions, internal delays, and processing and initial fuel values. The operations on the MannedSARPlatform control the flight route for the vehicle. It has the option to fly directly to the location the AutonomousSARUAV detected the person overboard, or it can fly to a predetermined location and await detection.

Behavior

The behavior of each system is described in separate state machine diagrams.

FishingVessel State Machine Diagram

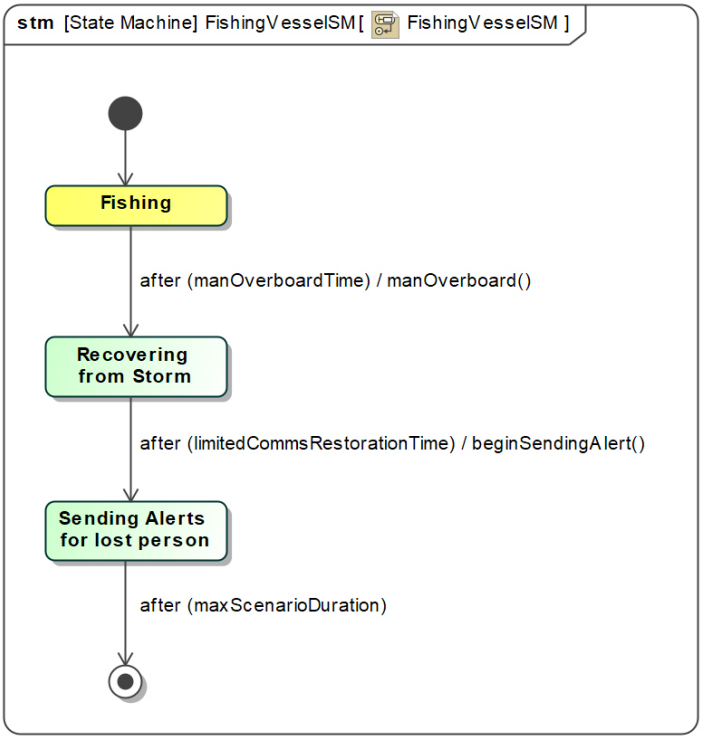

Figure 2. The fishing vessel's state machine diagram

The fishing vessel has the simplest and most straightforward state machine diagram of the systems in this mission. Initially, the vessel is fishing and operating as usual. At a given time, a person will fall overboard due to a storm, after which the vessel needs time to recover and restore communications. When communications are restored, the vessel will broadcast an emergency alert until the end of the mission.

LEODetectingConstellation State Machine Diagram

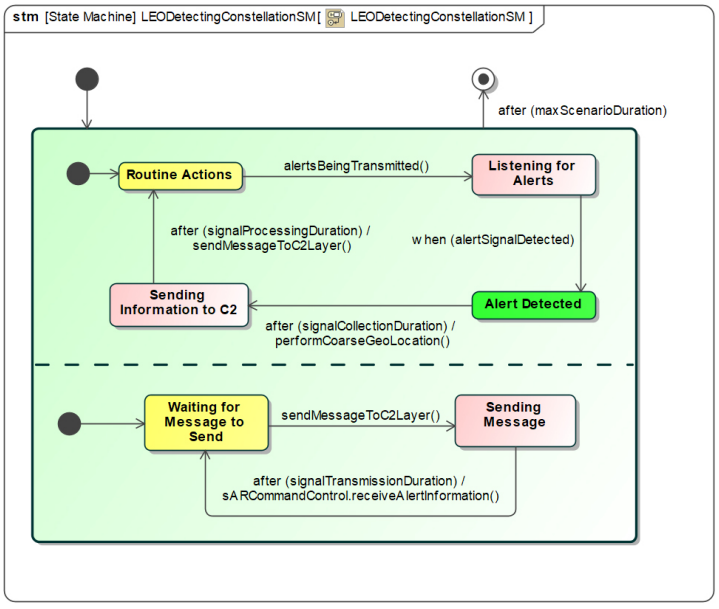

Figure 3. The LEO detecting constellation's state machine diagram

The main operating state of the LEODetectingConstellation has two orthogonal regions, meaning the regions separated by a dashed line are operating in parallel with respect to each other. The top region describes the primary function of the constellation. Initially the constellation performs routine actions until it is notified there are emergency alerts being transmitted. Then, it waits until it detects the alert. When the constellation detects the alert, there is a brief delay in collecting the signal before the satellite that detected the signal determines the initial area it received the signal from. There is another internal delay as the signal is processed by the satellite hardware, which takes place before the relevant information is sent to the command and control node.

The bottom state describes the process of sending the message to the C2 layer. Initially, the satellite is waiting for a message to be sent. After the information from the alert and initial search has been defined, that information is sent to the C2 node. There is a delay in signal transmission before the C2 node receives the information.

AutonomousSARUAV State Machine Diagram

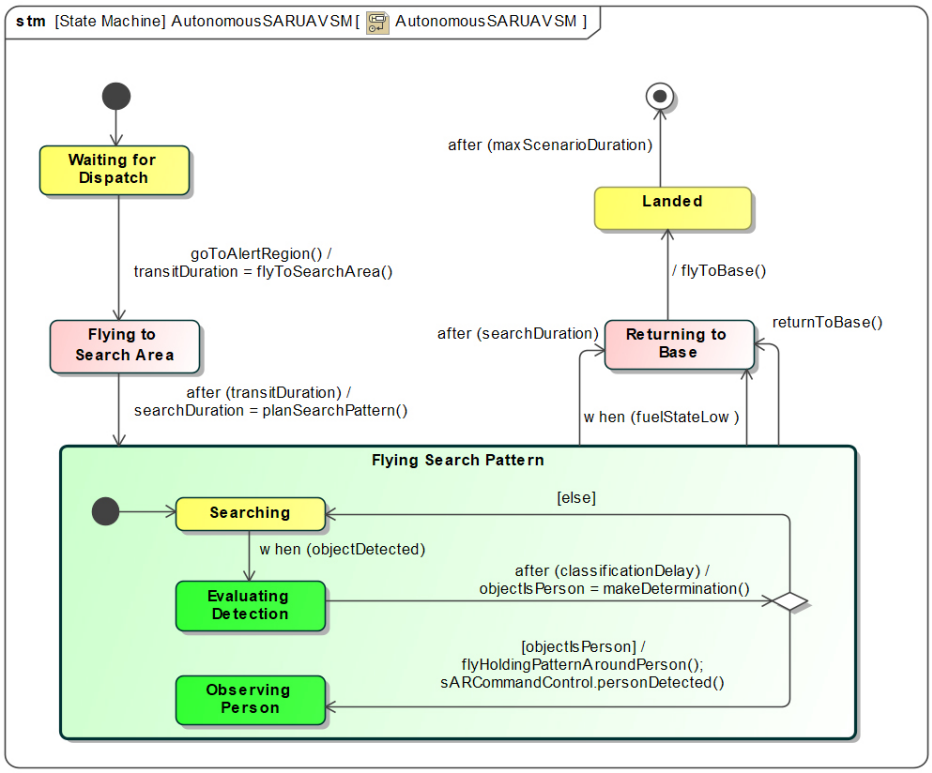

Figure 4. The autonomous UAV's state machine diagram

The starting state for the AutonomousSARUAV is an idle state, in which it is waiting to be dispatched to the search region. When the "goToAlertRegion" operation is called, the vehicle will begin flying to the search area. Upon arrival, it will plan the search pattern flight route based on the specified search values.

The state the UAV is in while flying the search pattern is a composite state, meaning it contains its own sequence of substates. While the UAV is flying the search pattern, it will attempt to detect objects using its sensing payload. When the UAV detects an object, it will determine if that object is a person and — based on that classification — whether it will continue searching or fly a holding pattern around the person while notifying the C2 node that a person has been located. This is shown in the state machine diagram as a choice block with a guard.

Multiple transitions can be used to show separate paths into or out of a state. This is demonstrated in the transitions between the "Flying Search Pattern" state and the "Returning to Base" state, in which this transition can happen due to low fuel, reaching the end of the search area, or being commanded back to base. After returning to base, the UAV will land, completing its mission.

MannedSARPlatform State Machine Diagram

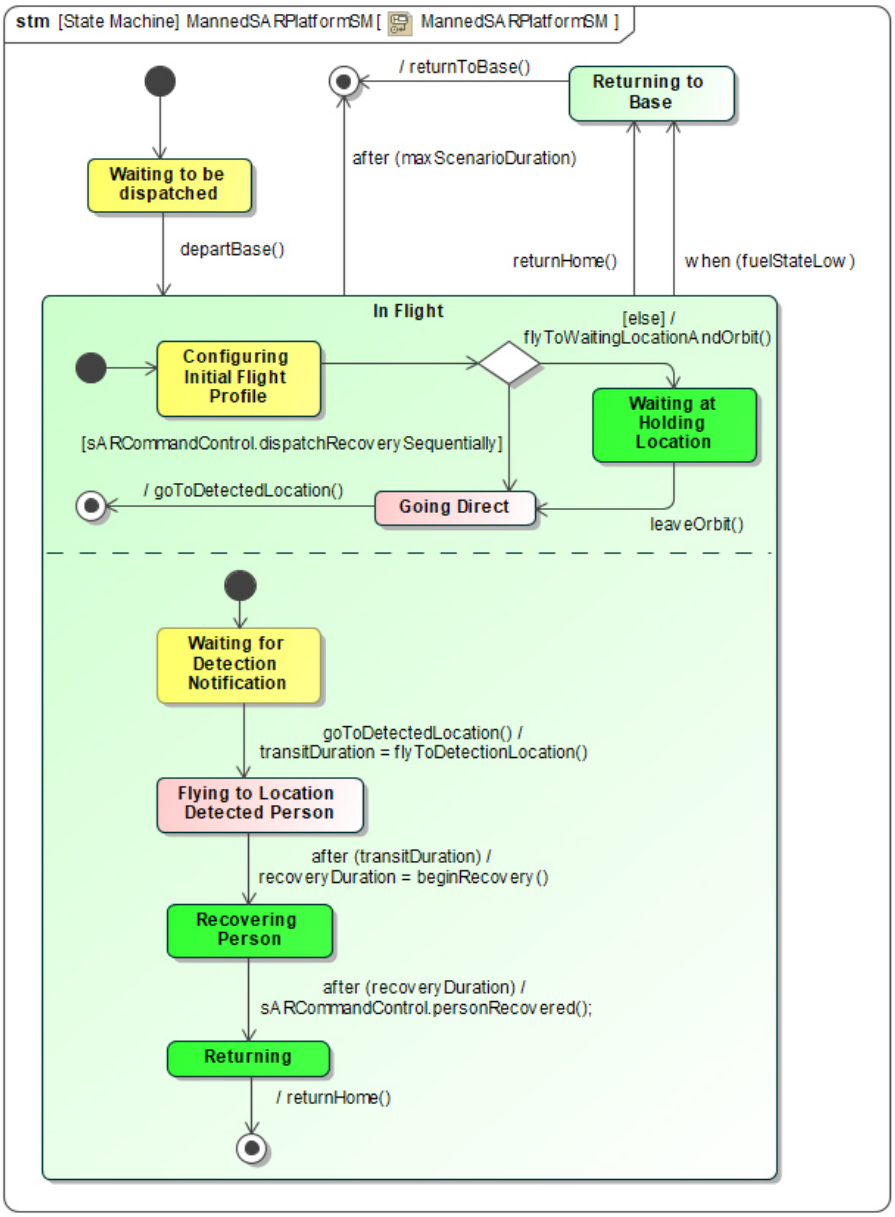

Figure 5. The crewed platform's state machine diagram

Like the AutonomousSARUAV, the MannedSARPlatform starts waiting to be dispatched by the C2 layer. After being instructed to take off and leave the base, it enters an orthogonal state block. The top region defines which recovery mode the crewed platform is operating in based on the "dispatchRecoverySequentially" property on the SARCommandControl block. The crewed platform will either fly directly to the detection location or wait at the holding location until the SARCommandControl instructs it to go to the detected location.

In the bottom region, the platform waits for a detection notification. When it is told to go to the detection location, it will fly to that point and begin the recovery process for the person overboard. After the process is completed, the MannedSARPlatform will alert the C2 node that the person has been recovered and return to base. The main "In Flight" state also has multiple triggers to return to base in case the crewed platform reaches a critically low fuel level before the mission is complete.

SARCommandControl State Machine Diagram

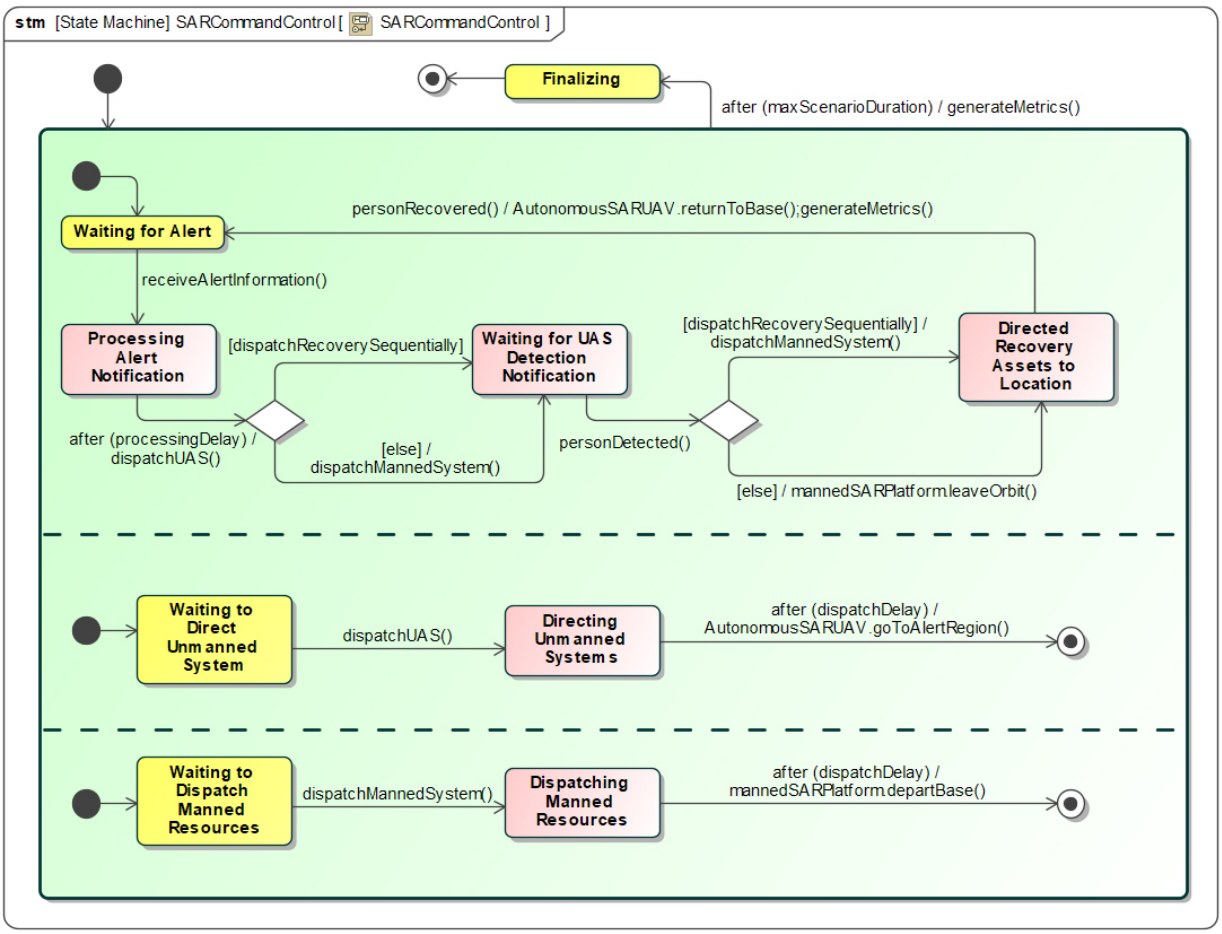

Figure 6. The command-and-control node's state machine diagram

The SARCommandControl has one large state with three orthogonal regions. The top region shows the process of coordinating the crewed and uncrewed systems throughout the mission, while the bottom two regions display the status of the SAR platform deployments.

In the top region, the SARCommandControl block waits for an alert from the satellite constellation. After processing the information from the alert, it dispatches the UAV and crewed recovery platform. The UAV is dispatched immediately, but if the SARCommandControl is operating in the sequential dispatch mode, the crewed recovery platform is not immediately dispatched.

Once the person is detected by the UAV, the C2 node will dispatch the crewed platform if it hasn't already. Otherwise, it will instruct the crewed platform to leave its holding orbit and fly to the detection location. Once the crewed platform has successfully recovered the person overboard, both systems will return to base.

The bottom two regions are separated between the UAV and crewed platform, but they function identically. Initially, the system is waiting to dispatch the given system. Once it is told to do so, there is a delay as the instruction is executed, then the system is dispatched for its role. These states are used to show the current dispatch status of the SAR platforms.

Instance Specification Diagram

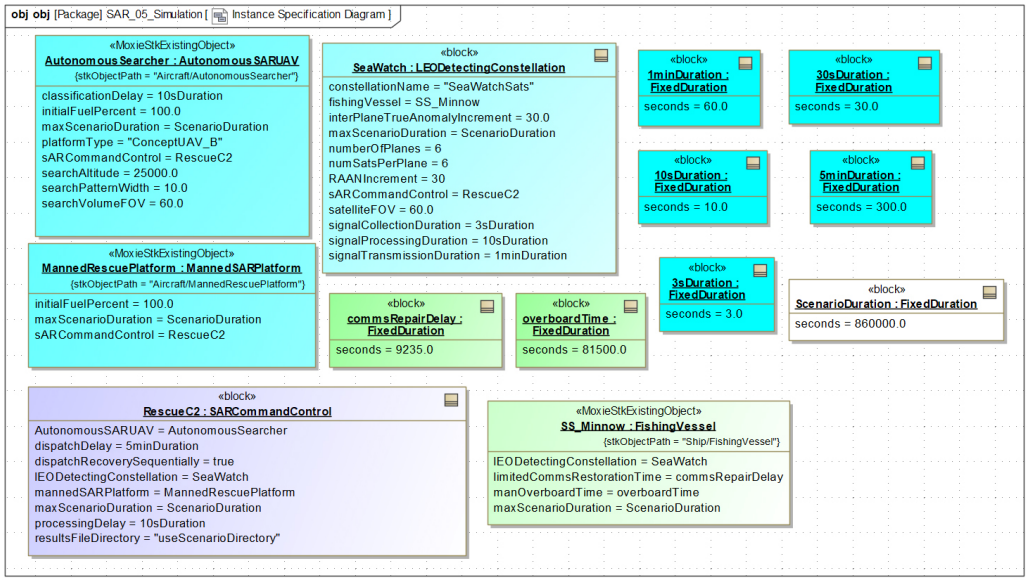

Figure 7. The instance specification diagram

The instance specification diagram holds the specifications for the systems used in the analysis. The value properties shown in the block definition diagram are given primitive values with data types like integers, strings, and Booleans, while part property values are slotted with references to other blocks. Each instance of a block is given a unique name to identify its usage in other blocks' properties.

Analysis Models

Finally, alongside the SysML model of the system architecture, there are physics-based simulation and analysis tools that can be used to validate the designs. This example used a commercially available mission analysis tool called Ansys Systems Tool Kit (STK). This tool enables a user to create simulated representations of each of the systems described above and execute the mission in a physics-based, coordinated environment. Performing precise and accurate discrete event detection for multiple cyber-physical and phenomenological systems requires carefully analyzing these simulated results together in a common operating environment. In this way, a user can understand the complex interactions between systems, evaluate individual system and component performance, and validate mission architecture in a wide range of operational scenarios.

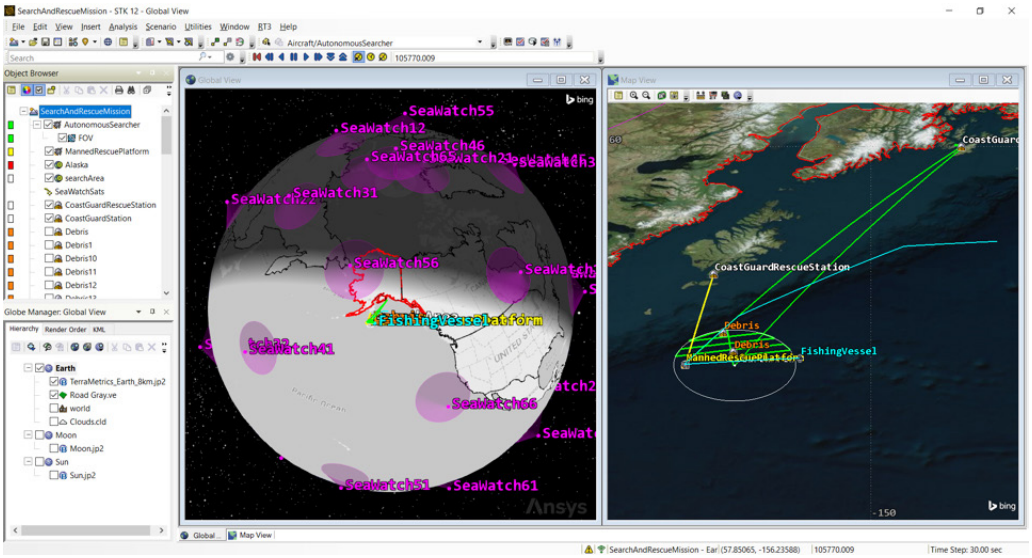

Figure 8. Ansys Systems Tool Kit (STK) performing mission simulation and analysis

/ Execution

Initialization

Linking the structure and behaviors described in the SysML models to the analytical representations of each system enables a user to identify emergent behaviors and validate the system architecture. To do this, block properties are connected to their analytical equivalents in the physics-based tools to ensure consistency across models, while operations are used to interact with the simulation during execution and pass data between tools.

For example, the “numberOfPlanes” and “numSatsPerPlane” value properties from the LEODetectingConstellation in the SysML can be directly connected to the same parameters used in STK when creating a satellite constellation. It is important to note that not every block property needs to be connected to an analytical parameter. Similarly, not every part of the analytical model will be represented in the SysML.

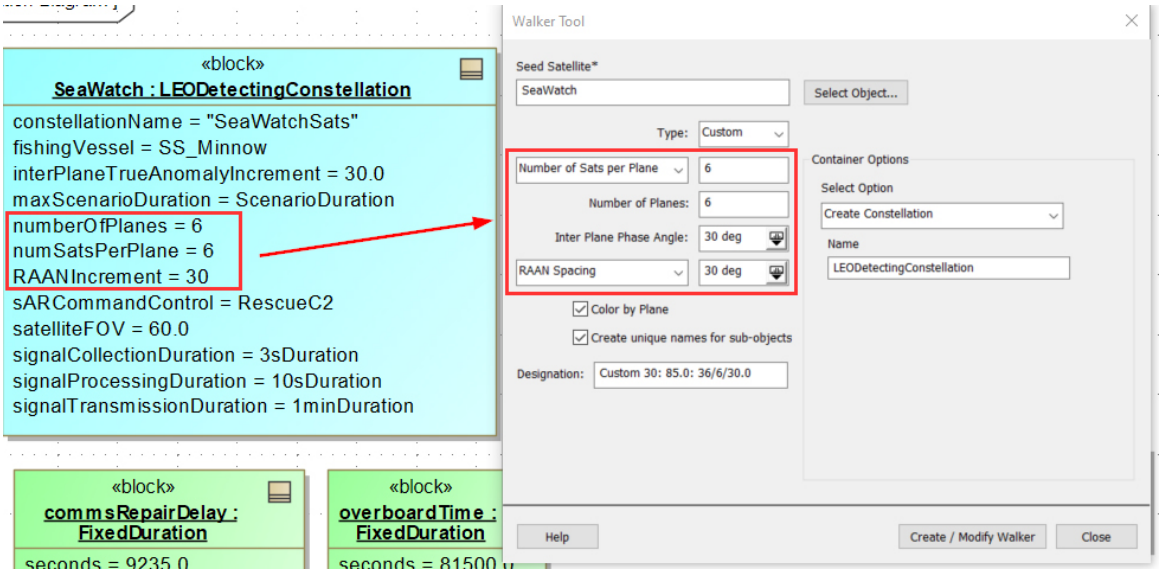

Figure 9. A direct link between value properties in the SysML and analytical parameters in STK

Operations are also linked to physics-based analysis. They can be used to execute some changes in the tools when called as the effect of a transition in the state machine diagrams. For example, the “flyToSearchArea” operation on the AutonomousSARUAV updates the vehicle’s flight route in STK to redirect the aircraft and begin its flight to the search area.

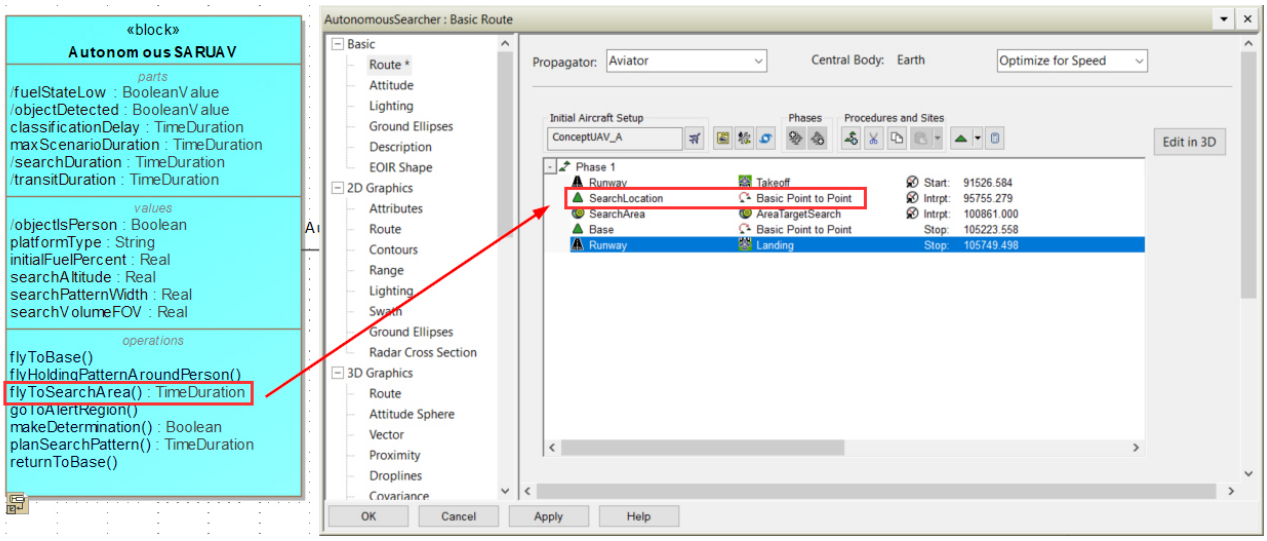

Figure 10. The effect of an operation is shown in the vehicle flight route in STK

Simulation

After the SysML model is initialized from the analytical tools, execution of the simulation can begin. This is done by initiating the simulation in each instance's state machine and traversing through the diagram as dictated by the transitions between states. On each diagram, when the trigger condition and the guard on the next transition is met, the effect of that transition executes and then the state machine transitions to the next state.

The timing for transitions between states can be modeled many ways. Physical systems may include models of clocks that tick at a specific rate or duty cycle. However, natural physical phenomena generally don't use clocks and operate continuously in time. To simulate system models accurately and precisely, it's important to synchronize the continuous and discrete timing systems together when detecting events. Physical systems may only generate events as a function of their discrete steps, while other phenomena can happen at any time between these "ticks." The model must be clear and correct on these distinctions, and the analysis tools are subsequently responsible for simulating the model accurately and precisely.

Each system starts in the initial state shown in its respective state machine diagram. Next, the transition triggers from each initial state are evaluated to determine which transition happens first relative to one another. In this example, the first event to occur is the person falling overboard in the FishingVessel state machine. This process continues until all state machines reach the final state or the end of the simulation is reached.

One example in which this evaluation is crucial for the simulation is in the AutonomousSARUAV state machine diagram (Figure 4). While the vehicle is in the "Flying Seach Pattern" composite state, there are three transitions that will cause the vehicle to return to base: the search pattern has been completed, the aircraft's fuel is too low to continue, or the vehicle is commanded to return to base. Even if the aircraft is minutes away from the detecting the man overboard, crossing the low-fuel threshold will cause it to return to base and the mission to fail.

The determination of when these events happen is delegated to the analytical simulation tools. The fuel burn rate of the UAV can be used by the analytical models to determine when the aircraft will run low on fuel. Likewise, the satellite constellation orbit parameters determine when a satellite will be overhead the fishing vessel to receive the distress signal. By propagating the satellite's location using numerical or analytical methods, the tool can provide information about when the detection of the signal will occur.

After finding when these transitions occur, the effect of those events is also executed in the analysis tools as the state machines progress. For example, in the MannedSARPlatform's state machine diagram (Figure 5), there is a choice block in the upper region of the main "In Flight" block. If the SARCommandControl's "dispatchRecoverySequentially" property value is true, the "flyToWaitingLocationAndHold" operation will be called. If the value is false, that operation will not be called. Running the simulation multiple times with different values for this property will produce flight routes in STK that reflect the different traversal of states in the state machine diagram.

While we have a direct connection between the system architecture and the tools for analysis, it's crucial that simulation-specific methodology is not included in the SysML model. This concept is essential for executing these models. By keeping a level of abstraction between the SysML model and the analysis done to execute the model, different analytical implementations can be changed out without significant change to the SysML. This creates a concept of operations (CONOPS) that can be executed with different tools at different fidelities as needed.

This is especially important in the context of the system life cycle. The system's design is continually changing, as is the type and fidelity of the modeling, simulation, and analysis tools used to validate the system designs. As the design is decomposed into more detail and lower-level components are introduced, the higher-fidelity analysis tools can be seamlessly integrated into the execution.

Validation

At the conclusion of the simulation, key results are tied back to the requirements in the model to verify a successful outcome. Very often, the behaviors described in the state machine diagrams do not lead to the desired outcome when executed as they are written. Implicit assumptions or complicated relationships with other system behaviors can produce unexpected results. Modifications to the system design can be made and immediately validated through this execution method, enabling large trade-space exploration and design trade-offs to be evaluated.

By modifying the initial conditions and values on the instance specification diagram, the mission can be executed in many operational conditions that the real system may face. These conditions can put a system in situations that stress its capabilities or inhibit its ability to communicate and coordinate. Evaluating the mission in this variety of scenarios ensures that the systems as designed will be able to meet overall requirements. It also increases the likelihood of identifying problematic situations that could jeopardize the mission.

Conclusion

Executing simulation and analysis directly from the SysML model for a system is crucial for validating engineering designs throughout the product life cycle. It enables simulations to be connected directly to the authoritative source(s) of truth while supporting the ability to interchange models as simulation complexity increases.

By using the appropriate physics models to evaluate events, perform analysis, and drive the simulation, system architecture can be validated against the mission. This enables system engineers and MBSE practitioners to properly assess measures of effectiveness and performance as the system interacts with other systems and the mission environment. This process was demonstrated through a notional search-and-rescue mission in this example, but it has been applied to real-world systems that are much more complex and detailed.

ANSYS, Inc.
Southpointe
2600 Ansys Drive
Canonsburg, PA 15317
U.S.A.
724-746-3304
ansysinfo@ansys.com

When visionary companies need to know how their world-changing ideas will perform, they close the gap between design and reality with Ansys simulation. For more than 50 years, Ansys software has enabled innovators across industries to push boundaries by using the predictive power of simulation. From sustainable transportation to advanced semiconductors, from satellite systems to life-saving medical devices, the next great leaps in human advancement will be powered by Ansys.

Ansys and any and all ANSYS, Inc. brand, product, service and feature names, logos and slogans are registered trademarks or trademarks of ANSYS, Inc. or its subsidiaries in the United States or other countries. All other brand, product, service and feature names or trademarks are the property of their respective owners.

Visit www.ansys.com for more information.

©2024 ANSYS, Inc. All rights reserved.